Technical Report

Extended Simple Colored Petri Nets

Antonio Camurri and Alessandro Coglio

Dept. of Informatics, Systems, and Telecommunications (DIST)
Faculty of Engineering – University of Genoa

Viale Causa 13
16145 Genova, Italy

E-mail: {music, tokamak}@dist.unige.it

December 1997

Extended Simple Colored Petri Nets

Antonio Camurri and Alessandro Coglio

Abstract

We present a new class of Petri Nets, called *Extended Simple Colored Petri Nets* (*ESCP-nets*), which are essentially Simple Colored Petri Nets (SCP-nets) with three added features: first, there is a built-in type of real numbers; second, tokens can be forced to wait in places; third, an interface specifies how the Net can be externally supervised. Even if these added features also add complexity, ESCP-nets still constitute a valuable trade-off between the simplicity of P-nets and the convenience of CP-nets. We have successfully employed ESCP-nets in an industrial project for Demag-Italimpianti (the largest Italian industry producing plants), consisting in the realization of software tools for the development and execution of plants built according to an architecture where an ESCP-net executor is supervised by means of declarative rules about the marking of the Net. Even if we have presently employed ESCP-nets in the field of plant simulation only, their characteristic might be well-suited to other fields as well.

1. Introduction

In this report we present a new class of Petri Nets, called *Extended Simple Colored Petri Nets* (*ESCP-nets*). As implied by their name, ESCP-nets constitute an extension of Simple Colored Petri Nets (SCP-nets) [3]. The extension substantially consists of three features. First, besides user-defined enumerative types there is a built-in (non-enumerative) type of real numbers, by which continuous (attributes of) entities can be modeled or specified. Second, tokens can be forced to wait (non-deterministic) amounts of time before being removed from a place, so that time durations can be modeled or specified. Third, an ESCP-net comes equipped with an interface through which an external supervisor can constrain the behavior of the Net.

The three above features have been inspired by the field of plant simulation, precisely that kind of simulation aimed at obtaining experimental data about the working of plants (e.g. throughputs, utilization percentages, buffer sizes, and so on). Real numbers can model continuous quantities, e.g. weights, geometrical sizes. Tokens waiting in places can model durations of processes, e.g. the machining of a manufacturing part. Decisions depending on the overall state of a plant (e.g. to optimally allocate resources) can be modeled by an external supervisor, which can constrain the behavior of an ESCP-net through the specified interface. ESCP-nets are in fact an extremely valuable tool for the development of plant simulators [2], because they can conveniently express most of the crucial aspects of plants (i.e. flows of materials and parts, concurrency and synchronization of processes, use of shared resources, etc.) in a high-level, clear, and readable way. We have successfully employed ESCP-nets in an industrial project for Demag-Italimpianti (the largest Italian industry producing plants), consisting in the realization of software tools for the development and execution of plant simulators built according to an architecture [1] where an executor of ESCP-nets [4] is supervised by means of declarative rules about the marking of the Net. By means of our tools, correct, readable, and easy-to-maintain simulators can be developed with dramatical reductions of development times and errors.

The advantages of ESCP-nets over “classical” Petri Nets (P-nets) [7, 6] and Colored Petri Nets [5] are substantially the same as those of SCP-nets: while providing

much more convenience than P-nets, ESCP-nets can be more easily implemented and seem to be more amenable to formal analysis than CP-nets. Even if, of course, their three added features (with respect to SCP-nets) also add complexity, ESCP-nets are still much less complex than CP-nets. In particular, no general programming language is needed to define types and operations over them, because operations upon enumerative items and real numbers can be easily expressed, for example, by integrating built-in operations over real numbers with argument-result enumerations (as we have done in our industrial software tools).

In Section 2 we formally present ESCP-nets. Such a presentation assumes knowledge of that of SCP-nets in [3] (anyway, the formal definitions we give here are self-contained, i.e. they are not based upon those for SCP-nets). In Section 3 we outline future work. The mathematical concepts and notations used in Section 2 are explained in the Appendix.

2. Formal Presentation

As we present formal definitions, we instantiate them to the very simple ESCP-net depicted in Figure 1, which models an artificial, abstract process where unpainted cubes and spheres (of various sizes) are painted with red, green, or blue paint, by means of suitable subtractive color syntheses of cyan, magenta, and yellow paint.

2.1 Topology

The topology of an ESCP-net is defined exactly like that of an SCP-net. There are places, transitions, and arcs connecting them.

Definition 1. A *topology* is a quadruple

$$\boldsymbol{T} = \langle P, T, A, \chi \rangle$$

where:
(1) P is a finite set whose elements are called *places*;
(2) T is a finite set whose elements are called *transitions*;
(3) A is a finite set whose elements are called *arcs*;
(4) $P \cap T = \varnothing$;
(5) $\chi \in [\, A \to (P \times T) \cup (T \times P)\,]$ and is called *connection*.
For each $a \in A$, let $p \in P$ and $t \in T$ such that $\chi(a) = \langle p, t \rangle$ or $\chi(a) = \langle t, p \rangle$; we define:
(1) $\chi_P(a) = p$;
(2) $\chi_T(a) = t$.
For each $x \in P \cup T$, we define the sets $In(x)$ and $Out(x)$ of the *incoming arcs* and *outgoing arcs* of x as follows:
(1) $In(x) = \{\, a \in A \mid \exists\, y \in P \cup T : \chi(a) = \langle y, x \rangle \,\}$;
(2) $Out(x) = \{\, a \in A \mid \exists\, y \in P \cup T : \chi(a) = \langle x, y \rangle \,\}$.
For each $p \in P$ and $t \in T$, we say that:
(1) p is an *input place* of t and t is an *output transition* of p iff $In(t) \cap Out(p) \neq \varnothing$;
(2) p is an *output place* of t and t is an *input transition* of p iff $In(p) \cap Out(t) \neq \varnothing$.

In Figure 1 the topology consists of five places (UNPAINTED, PAINTING, PAINTED, TANKS1, and TANKS2), two transitions (START[1] and END), and eight arcs (whose names have been left unspecified to reduce graphical cluttering), with the connection defined as graphically shown.

[1] For now, ignore its shading.

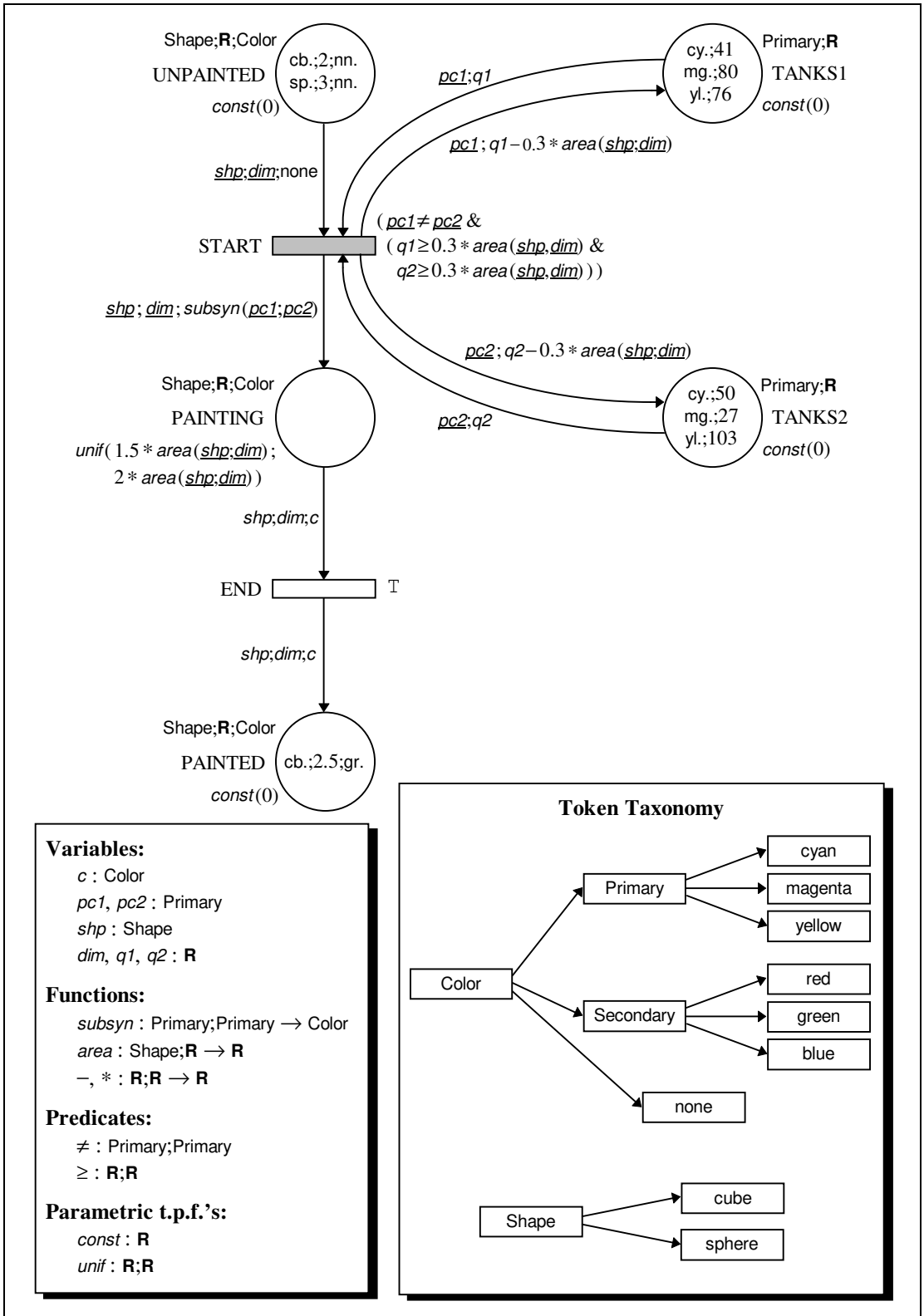

Figure 1: A simple example of ESCP-net.

2.2 Token Taxonomy

As mentioned in Section 1, each ESCP-net has, besides user-defined enumerative types, a pre-defined built-in type of real numbers (which is of course non-enumerative). More precisely, the distinguished symbol **R** is a base type of any ESCP-net. **R** has no

super-types or sub-types, and its base tokens are all the real numbers[2]. **R** and real numbers can be freely concatenated with user-defined base types and base tokens, to make up types and tokens. We in fact give the following definition.

Definition 2. A *token taxonomy* $\boldsymbol{X}$ is a finite non-empty DAG of symbols distinct from **R** and **E**, such that no symbol is isolated in the DAG.
The set Y_B of *base types* is

$$Y_B = \mathrm{NTerm}(\boldsymbol{X}) \cup \{\mathbf{R}\}.$$

For each $y, y' \in Y_B$, we say that y is a *super-type* of y' and that y' is a *sub-type* of y iff

$$\mathbf{R} \notin \{y, y'\} \ \wedge \ y \rightarrow_{\boldsymbol{X}} y'.$$

The set Y of *types* is the smallest set such that

$$Y = Y_B \cup \{\ y;y' \text{ string} \mid y, y' \in Y\ \}.$$

For each $y, y' \in Y$, we say that y is a *super-type* of y' and that y' is a *sub-type* of y iff for some $y_1, \ldots, y_n, y_1', \ldots, y_n' \in Y_B$ we have:
(1) $y = y_1;\ldots;y_n \ \wedge \ y' = y_1';\ldots;y_n'$;
(2) $\forall\, i \in \{1,\ldots,n\} : (\ y_i = y_i' \vee y_i$ is a super-type of $y_i'\ )$;
(3) $\exists\, i \in \{1,\ldots,n\} : y_i \neq y_i'$.
The family $\{K_y\}_{y \in Y}$ of *tokens* is the smallest family such that:
(1) $\forall\, y \in Y_B - \{\mathbf{R}\} : K_y = \{\ k \mid k \in \mathrm{Term}(\boldsymbol{X}) \wedge y \rightarrow_{\boldsymbol{X}} k\ \}$;
(2) $K_{\mathbf{R}} = \mathbf{R}$;
(3) $\forall\, y, y' \in Y : K_{y;y'} = \{\ k;k' \text{ string} \mid k \in K_y \wedge k' \in K_{y'}\ \}$.
The set K of all tokens is thus

$$K = \bigcup_{y \in Y} K_y\ .$$

The set K_B of *base tokens* is

$$K_B = \bigcup_{y \in Y_B} K_y\ .$$

For each $k \in K$ and $y \in Y$, we say that k *has type* y iff

$$k \in K_y.$$

The token taxonomy $\boldsymbol{X}$ only specifies user-defined base types and base tokens. Therefore, $\boldsymbol{X}$ must not contain the distinguished symbol **R** to avoid ambiguities[3]. Note that real numbers are used in token strings as if they were symbols. In fact, with slight abuse, we really regard the base tokens of type **R** as symbols in bijective correspondence with real numbers (and not as real numbers themselves).

In Figure 1 there is a type Primary with tokens cyan, magenta, and yellow, and a type Secondary with tokens red, green, and blue[4]. Color is a super-type of both Primary and Secondary, and its tokens are the six above plus none. There is also a type Shape with tokens cube and sphere. Other examples of types are Primary;**R** and Shape;**R**;Color; for instance, Color;**R** is a super-type of Primary;**R**. Other examples of tokens are cyan;41 (which can represent 41 liters of cyan paint), of type Primary;**R** and Color;**R**, and cube;2.5;green (which can represent a green cube whose edge measures 2.5 meters), of types Shape;**R**;Secondary and Shape;**R**;Color.

---

[2] However, all our concepts can be straightforwardly adapted to cases of practical implementations, where the base tokens of **R** are only a finite subset of **R** (e.g. all the floating point numbers representable in a given 32-bit format).
[3] The reason why it is also required that the symbol **E** does not appear in $\boldsymbol{X}$ will be clarified below.
[4] In subtractive color synthesis, cyan, magenta, and yellow are primary colors, while red, green, and blue are secondary colors.

2.3 Time Probability Function

As we will see below, the amount of time a token must wait in a place is randomly generated according to a time probability function (t.p.f.)[5], which assigns a probability value to each natural number, where natural numbers represent discrete time durations in a given unit. The concept of t.p.f. is formalized as follows.

Definition 3. A *time probability function* (*t.p.f.*) is a family

$$\{\pi_i\}_{i\in\mathbf{N}}$$

of real numbers such that:
(1) $\forall\ i \in \mathbf{N} : 0 \le \pi_i \le 1$;
(2) $\sum_{i\in\mathbf{N}} \pi_i = 1$.
Π is the set of all t.p.f.'s.

Of course, each π_i is the probability of the time duration $i \in \mathbf{N}$, and their total sum must yield 1.

As we will see, in Figure 1 we make use of t.p.f.'s Unif(n_1, n_2), with $n_1, n_2 \in \mathbf{N}$ and $n_1 \le n_2$, characterized by $\pi_i = 1 / (n_2 - n_1 + 1)$ for all i such that $n_1 \le i \le n_2$, and $\pi_i = 0$ for all i such that $i < n_1$ or $i > n_2$. So, a t.p.f. Unif(n_1, n_2) can be used to generate a uniformly random time duration between n_1 and n_2 (inclusive). In Figure 1 we also make use of t.p.f.'s Const(n), with $n \in \mathbf{N}$, characterized by $\pi_n = 1$, and $\pi_i = 0$ for all $i \ne n$. So, a t.p.f. Const(n) can be used generate a deterministic (i.e. constant) time duration n.

2.4 Signature

Signatures of ESCP-nets are like those of SCP-nets, but in addition to variables, functions, and predicates, they also specify symbols denoting parametric t.p.f.'s, i.e. functions mapping tokens to t.p.f.'s. Parametric t.p.f.'s, as we will see, provide the base of our very flexible generation of waiting times, which allows different tokens to wait different amounts of time in a same place.

Definition 4. Given a token taxonomy $\boldsymbol{X}$, a *signature* for $\boldsymbol{X}$ is a quadruple

$$\boldsymbol{S} = \langle \{V_y\}_{y\in Y}, \{F_{y,y'}\}_{y,y'\in Y}, \{R_y\}_{y\in Y}, \{H_y\}_{y\in Y} \rangle$$

where:
(1) $\{V_y\}_{y\in Y}$ is a family of pairwise disjoint sets of symbols called *variables*;
(2) $\{F_{y,y'}\}_{y,y'\in Y}$ is a family of pairwise disjoint sets of symbols called *functions*;
(3) $\{R_y\}_{y\in Y}$ is a family of pairwise disjoint sets of symbols called *predicates*;
(4) $\{H_y\}_{y\in Y}$ is a family of pairwise disjoint sets of symbols called *parametric t.p.f.'s*;
(5) $\left(\bigcup_{y\in Y} V_y \cup \bigcup_{y,y'\in Y} F_{y,y'} \cup \bigcup_{y\in Y} R_y \cup \bigcup_{y\in Y} H_y\right)$ is finite.

The sets V, F, R, and H of all variables, functions, predicates, and parametric t.p.f.'s are thus

$$V = \bigcup_{y\in Y} V_y\ ,\quad F = \bigcup_{y,y'\in Y} F_{y,y'}\ ,\quad R = \bigcup_{y\in Y} R_y\ ,\quad \text{and}\quad H = \bigcup_{y\in Y} H_y\ .$$

A type is associated to each parametric t.p.f.

In Figure 1 there are seven variables (*c*, *pc1*, *pc2*, *shp*, *dim*, *q1*, and *q2*), four

[5] T.p.f.'s and parametric t.p.f.'s (see below) were respectively called *time probability distributions* and *distribution families* in the informal description of ESCP-nets we gave in [2]. Anyway, *t.p.f.* and *parametric t.p.f.* are the preferred terms and are in fact used in formal definitions here.

functions (*subsyn*, *area*, −, and ∗), two predicates (≠ and ≥), and two parametric t.p.f.'s (*const* and *unif*), with the indicated associated types.

2.5 Interpretation

An interpretation specifies the semantics of functions, predicates, and parametric t.p.f.'s of a signature. The semantics of a parametric t.p.f. maps tokens of the associated type to t.p.f.'s, while the semantics of a function or predicate is analogous to that in an SCP-net. However, there is one difference. Since many operations involving real numbers are partial (e.g. division is not defined if the divisor is null), we explicitly represent partialness by allowing the semantics of a function, predicate, or parametric t.p.f. to return, upon certain tokens (containing real numbers or not, for uniformity), the distinguished symbol **E** (which, as required in Definition 2, must not appear in the token taxonomy to avoid ambiguities with base tokens), which denotes an "error". We therefore give the following definition.

Definition 5. Given a token taxonomy ***X*** and a signature ***S*** for ***X***, an *interpretation* for ***X*** and ***S*** is a triple

$$\boldsymbol{I} = \langle \phi, \rho, \eta \rangle$$

where:
(1) $\phi \in [\, F_{y,y'} \to [K_y \to K_{y'} \cup \{\mathbf{E}\}]\,]_{y,y' \in Y}$ and is called *function interpretation*;
(2) $\rho \in [\, R_y \to [K_y \to \mathbf{B} \cup \{\mathbf{E}\}]\,]_{y \in Y}$ and is called *predicate interpretation*;
(3) $\eta \in [\, H_y \to [K_y \to \Pi \cup \{\mathbf{E}\}]\,]_{y \in Y}$ and is called *parametric t.p.f. interpretation*.

In Figure 1, the semantics of *subsyn* models the subtractive color synthesis of two primary colors, as follows:

$$\begin{aligned}
&\phi(\textit{subsyn})(\text{cyan};\text{cyan}) = \text{cyan}\ ,\\
&\phi(\textit{subsyn})(\text{cyan};\text{magenta}) = \text{blue}\ ,\\
&\phi(\textit{subsyn})(\text{cyan};\text{yellow}) = \text{green}\ ,\\
&\phi(\textit{subsyn})(\text{magenta};\text{cyan}) = \text{blue}\ ,\\
&\phi(\textit{subsyn})(\text{magenta};\text{magenta}) = \text{magenta}\ ,\\
&\phi(\textit{subsyn})(\text{magenta};\text{yellow}) = \text{red}\ ,\\
&\phi(\textit{subsyn})(\text{yellow};\text{cyan}) = \text{green}\ ,\\
&\phi(\textit{subsyn})(\text{yellow};\text{magenta}) = \text{red}\ ,\\
&\phi(\textit{subsyn})(\text{yellow};\text{yellow}) = \text{yellow}\ .
\end{aligned}$$

The semantics of *area* computes the surface area of a cube or sphere, as follows (*pi* ∈ **R** is the ratio of a circumference measure to its diameter):

$$\begin{aligned}
&\forall\, x \in \mathbf{R} : \phi(\textit{area})(\text{cube};x) = 6 \cdot x^2,\\
&\forall\, x \in \mathbf{R} : \phi(\textit{area})(\text{sphere};x) = 4 \cdot \textit{pi} \cdot x^2.
\end{aligned}$$

The semantics of − and ∗ are respectively subtraction and multiplication of real numbers. The semantics of ≠ is inequality of primary colors. The semantics of ≥ is the greater-than-or-equal-to relation over real numbers. The semantics of *const* maps a real number to the deterministic t.p.f. of the integer value obtained by rounding it, provided such a value is non-negative:

$$\forall\, x \in \mathbf{R} : \eta(\textit{const})(x) = \textbf{if}\ (\text{Round}(x) \geq 0)\ \textbf{then}\ \text{Const}(\text{Round}(x))\ \textbf{else}\ \mathbf{E}.$$

Finally, the semantics of *unif* substantially maps two real numbers to the uniform t.p.f. between the integer values obtained by rounding them, provided such values are non-negative and the first is not greater than the second:

$\forall\, x_1, x_2 \in \mathbf{R} : \eta(\mathit{unif})(x_1;x_2) =$ **if** $(0 \le \text{Round}(x_1) \le \text{Round}(x_2))$
then $\text{Unif}(\text{Round}(x_1), \text{Round}(x_2))$ **else** $\mathbf{E}$.

2.6 Expressions

Expressions for ESCP-nets are built in close analogy with those for SCP-nets.

Definition 6. Given a token taxonomy $\boldsymbol{X}$ and a signature $\boldsymbol{S}$ for $\boldsymbol{X}$, the family $\{E_y\}_{y \in Y}$ of *expressions* over $\boldsymbol{X}$ and $\boldsymbol{S}$ is the smallest family such that:
(1) $\forall\, y \in Y : K_y \cup V_y \subseteq E_y$;
(2) $\forall\, y, y' \in Y : \{\, f(e) \text{ string} \mid f \in F_{y,y'} \wedge e \in E_y \,\} \subseteq E_{y'}$;
(3) $\forall\, y, y' \in Y : \{\, e;e' \text{ string} \mid e \in E_y \wedge e' \in E_{y'} \,\} \subseteq E_{y;y'}$;
(4) $\forall\, y, y' \in Y$, y super-type of $y' : E_{y'} \subseteq E_y$.
The set E of all expressions is thus

$$E = \bigcup_{y \in Y} E_y \,.$$

For each $e \in E$ and $y \in Y$, we say that *e has type y* iff

$$e \in E_y.$$

We define the function *Var* over E as follows:
(1) $\forall\, k \in K : \mathit{Var}(k) = \varnothing$;
(2) $\forall\, v \in V : \mathit{Var}(v) = \{v\}$;
(3) $\forall\, f(e) \in E : \mathit{Var}(f(e)) = \mathit{Var}(e)$;
(4) $\forall\, e;e' \in E : \mathit{Var}(e;e') = \mathit{Var}(e) \cup \mathit{Var}(e')$.

Examples of expressions appearing in Figure 1[6] are *shp*;*dim*;*subsyn*(*pc1*;*pc2*), of type Shape;**R**;Color, as well as ∗(0.3;*area*(*shp*;*dim*)) (for ∗ we use a kind of infix notation to improve clarity), of type **R**.

2.7 Guards

Guards for ESCP-nets are also built in close analogy with those for SCP-nets.

Definition 7. Given a token taxonomy $\boldsymbol{X}$ and a signature $\boldsymbol{S}$ for $\boldsymbol{X}$, the set G of *guards* over $\boldsymbol{X}$ and $\boldsymbol{S}$ is the smallest set such that:
(1) $\texttt{TRUE} \in G$;
(2) $\forall\, y \in Y : \{\, r(e) \text{ string} \mid r \in R_y \wedge e \in E_y \,\} \subseteq G$;
(3) $\{\, (\sim g) \text{ string} \mid g \in G \,\} \subseteq G$;
(4) $\{\, (g_1 \,\&\, g_2) \text{ string} \mid g_1, g_2 \in G \,\} \subseteq G$.
We define the function *Var* over G as follows:
(1) $\mathit{Var}(\texttt{TRUE}) = \varnothing$;
(2) $\forall\, r(e) \in G : \mathit{Var}(r(e)) = \mathit{Var}(e)$;
(3) $\forall\, (\sim g) \in G : \mathit{Var}(\sim g) = \mathit{Var}(g)$;
(4) $\forall\, (g_1 \,\&\, g_2) \in G : \mathit{Var}(g_1 \,\&\, g_2) = \mathit{Var}(g_1) \cup \mathit{Var}(g_2)$.

Examples of guards appearing in Figure 1 are TRUE, ≠(*pc1*;*pc2*), as well as (≥(*q1*;∗(0.3;*area*(*shp*;*dim*))) & ≥(*q2*;∗(0.3;*area*(*shp*;*dim*)))) (for ≠ and ≥ we use a kind of infix notation to improve clarity).

[6] For now, ignore the underlining of some variable occurrences in them.

2.8 Stochastic Times

Expressions and guards are syntactical entities respectively evaluating to tokens and booleans. Analogously, stochastic times are syntactical entities evaluating to t.p.f.'s. They are simply built out of expressions and parametric t.p.f.'s, as formalized below.

Definition 8. Given a token taxonomy $\boldsymbol{X}$ and a signature $\boldsymbol{S}$ for $\boldsymbol{X}$, the set ST of *stochastic times* over $\boldsymbol{X}$ and $\boldsymbol{S}$ is

$$ST = \{\ h(e) \mid h \in H_y \wedge e \in E_y\ \}.$$

We define the function *Var* over ST as follows:

$$\forall\ h(e) \in ST : Var(h(e)) = Var(e).$$

The function *Var* collects all the variables occurring in a stochastic time.

Examples of stochastic times appearing in Figure 1 are *const*(0), as well as *unif*(∗(1.5;*area*(*shp*;*dim*));∗(2;*area*(*shp*;*dim*))).

2.9 Binding and Evaluation

The concepts of binding and evaluation in ESCP-nets are analogous to those in SCP-nets. The difference is that also stochastic times must be evaluated, and that errors must be taken into account.

Definition 9. Given a token taxonomy $\boldsymbol{X}$, a signature $\boldsymbol{S}$ for $\boldsymbol{X}$, and an interpretation $\boldsymbol{I}$ for $\boldsymbol{X}$ and $\boldsymbol{S}$, let $V' \subseteq V$ be a set of variables; a *binding* for V' is a function

$$\beta \in [\ (V_y \cap V') \to K_y\ ]_{y \in Y}.$$

Let $E' \subseteq E$ be the largest subset of E such that $\bigcup_{e \in E'} Var(e) \subseteq V'$; the *evaluation* of E' with β is defined as follows:

(1) $\forall\ k \in K : \beta(k) = k$;

(2) $\forall\ f(e) \in E' : \beta(f(e)) =$ **if** $(\beta(e) =$ **E**$)$ **then E else** $\phi(f)(\beta(e))$;

(3) $\forall\ e;e' \in E' : \beta(e;e') =$ **if** $(\beta(e) =$ **E** $\vee\ \beta(e') =$ **E**$)$ **then E else** $\beta(e);\beta(e')$.

Let $G' \subseteq G$ be the largest subset of G such that $\bigcup_{g \in G'} Var(g) \subseteq V'$; the *evaluation* of G' with β is defined as follows:

(1) $\beta($TRUE$) =$ T;

(2) $\forall\ r(e) \in G' : \beta(r(e)) =$ **if** $(\beta(e) =$ **E**$)$ **then E else** $\rho(r)(\beta(e))$;

(3) $\forall\ (\sim g) \in G' : \beta(\sim g) =$ **if** $(\beta(g) =$ **E**$)$ **then E else if** $(\beta(g) =$ F$)$ **then** T **else** F;

(4) $\forall\ (g_1\ \&\ g_2) \in G' : \beta(g_1\ \&\ g_2) =$ **if** $(\beta(g_1) \in \{$F, **E**$\})$ **then** $\beta(g_1)$ **else** $\beta(g_2)$.

Let $ST' \subseteq ST$ be the largest subset of ST such that $\bigcup_{st \in ST'} Var(st) \subseteq V'$; the *evaluation* of ST' with β is defined as follows:

$$\forall\ h(e) \in ST' : \beta(h(e)) = \textbf{if}\ (\beta(e) = \textbf{E})\ \textbf{then E else}\ \eta(h)(\beta(e)).$$

An expression evaluates to **E** iff the semantics of some function occurring in it returns **E**. In other words, errors are "propagated" from sub-expressions to the whole expression.

A guard evaluates to **E** only if some expression occurring in it does or the semantics of some predicate occurring in it returns **E**. Note however that the converse does not hold, because of the way conjunctions are evaluated: if the first conjunct evaluates to F, the conjunction evaluates to F even if the second conjunct evaluates to **E**. This short-circuited evaluation is very useful for a guard expressing a condition such as $(x \neq 0 \wedge 20/x > 7)$, which yields F and not **E** in case $x = 0$.

A stochastic time evaluates to **E** iff its expression does or the semantics of its

parametric t.p.f. returns **E**.

For instance, in the ESCP-net in Figure 1, if a binding β is such that

$$\begin{aligned}&\beta(pc1) = \text{magenta},\\ &\beta(pc2) = \text{yellow},\\ &\beta(shp) = \text{cube},\\ &\beta(dim) = 2,\\ &\beta(q1) = 80,\\ &\beta(q2) = 103,\end{aligned}$$

we have that

$$\begin{aligned}&\beta(subsyn(pc1;pc2)) = \text{red},\\ &\beta(area(shp;dim)) = 24,\\ &\beta(q1 \geq 0.3 * area(shp;dim)) = \text{T},\\ &\beta(unif\,(1.5 * area(shp;dim)\ ;\ 2 * area(shp;dim)) = \text{Unif}\,(36, 48),\\ &\beta(q2 - 100) = 3\\ &\beta(const(-6)) = \mathbf{E}.\end{aligned}$$

2.10 Token System

The concept of token system of an ESCP-net is closely analogous to that of an SCP-net.

Definition 10. A *token system* is a triple

$$\boldsymbol{K} = \langle \boldsymbol{X}, \boldsymbol{S}, \boldsymbol{I} \rangle$$

where:
(1) $\boldsymbol{X}$ is a token taxonomy;
(2) $\boldsymbol{S}$ is a signature for $\boldsymbol{X}$;
(3) $\boldsymbol{I}$ is an interpretation for $\boldsymbol{X}$ and $\boldsymbol{S}$.

2.11 Labeling

Like an SCP-net, in an ESCP-net each place is labeled by a type, each arc by an expression, and each transition by a guard. In addition, each place is also labeled by a stochastic time, used to determine how much time tokens added to the place by some firing must wait before they can be removed from the place (as we will explain below).

Definition 11. Given a topology $\boldsymbol{T}$ and a token system $\boldsymbol{K}$, a *labeling* of $\boldsymbol{T}$ with $\boldsymbol{K}$ is a quadruple

$$\boldsymbol{L} = \langle \psi, \varepsilon, \gamma, \tau \rangle$$

where:
(1) $\psi \in [P \to Y]$ and is called *type function*;
(2) $\varepsilon \in [\ A.a \to E_{\psi(\chi_P(a))}\ ]$ and is called *expression function*;
(3) $\gamma \in [T \to G]$ and is called *guard function*;
(4) $\tau \in [P \to ST]$ and is called *time function*.
For each $t \in T$, we define

$$Var(t) = \bigcup_{a \in In(t) \cup Out(t)} Var(\varepsilon(a)) \cup \bigcup_{a \in Out(t)} Var(\tau(\chi_P(a))) \cup Var(\gamma(t)).$$

Analogously to SCP-nets, the function *Var* collects all the variables surrounding the transition. Note that also the variables occurring in stochastic times of output places are included. In fact, as we will see in Definition 16, the effects of a transition

firing is also determined by such stochastic times.

The types, stochastic times, expressions, and guards labeling places, arcs, and transitions in Figure 1 are indicated near the corresponding circles, arrows, and rectangles.

2.12 Control Interface

As mentioned in Section 1, an ESCP-net comes equipped with an interface through which its behavior can be constrained by an external supervisor. This interface consists in some transitions tagged as "controlled", and some of their surrounding variables also tagged as "controlled". The meaning is that an external supervisor has the capability of allowing or forbidding the firing of each controlled transition, and, in the former case, it also has the capability of designating which tokens can be assigned to the controlled variables. The concept of control interface formally specifies which are the controlled transitions and variables of an ESCP-net.

Definition 12. Given a topology $\boldsymbol{T}$, a token system $\boldsymbol{K}$, and a labeling $\boldsymbol{L}$ of $\boldsymbol{T}$ with $\boldsymbol{K}$, a *control interface* for $\boldsymbol{T}$, $\boldsymbol{K}$, and $\boldsymbol{L}$ is a pair

$$\boldsymbol{C} = \langle CT, \xi \rangle$$

where:

(1) $CT \subseteq T$ is a set of transitions called *controlled transitions*;

(2) $\xi \in [\, CT.t \to \mathcal{P}_\omega(Var(t)) \,]$; for each $t \in CT$, the variables in $\xi(t)$ are called *controlled variables* of t.

In Figure 1, the shading of START indicates that it is a controlled transition, and the underlining of the surrounding occurrences of *shp*, *dim*, *pc1*, and *pc2* indicates that they are controlled variables. This means that an external supervisor has the capability of deciding when a new painting process can start (by allowing START to fire), which of the unpainted cubes or spheres present in UNPAINTED must be painted (by designating suitable tokens to be assigned to *shp* and *dim*), and which primary color tanks are used (by designating suitable tokens to be assigned to *pc1* and *pc2*).

2.13 Extended Simple Colored Petri Net

As expected, an ESCP-net consists of a topology, a token system, a labeling, and a control interface.

Definition 13. An *Extended Simple Colored Petri Net* (*ESCP-net*) is a quadruple

$$\boldsymbol{ESCPN} = \langle \boldsymbol{T}, \boldsymbol{K}, \boldsymbol{L}, \boldsymbol{C} \rangle$$

where:

(1) $\boldsymbol{T}$ is a topology;

(2) $\boldsymbol{K}$ is a token system;

(3) $\boldsymbol{L}$ is a labeling of $\boldsymbol{T}$ with $\boldsymbol{K}$;

(4) $\boldsymbol{C}$ is a control interface for $\boldsymbol{T}$, $\boldsymbol{K}$ and $\boldsymbol{L}$;

(5) $\forall\, t \in T : \forall\, v \in Var(t) :\ (t \in CT \wedge v \in \xi(t))\ \vee$
$(\exists\, a \in In(t) : \exists\, e, e' \in E : \varepsilon(a) \in \{\ v,\ e;v,\ v;e,\ e;v;e'\ \}).$

It is required that, for each transition, each surrounding variable either is controlled (thus implying that the transition is also controlled) or constitutes, concatenated with zero or more other expressions, the expression labeling some incoming arc of the transition. This assures that, provided external supervision designates finitely many bindings for controlled variables, each transition is enabled

with only a finite number of bindings (see Definition 16), because each place is always marked by finitely many tokens (see Definition 15). For instance, if a (non-controlled) variable of type **R** only appeared as the expression labeling an outgoing arc of a transition, there might be infinitely many bindings, one for each real number[7]. The requirement also allows, given bindings for controlled variables from external supervision, an easy computation of the bindings with which transitions are enabled, without the need of inverting any function. For instance, if a (non-controlled) variable only appeared as "argument expression" of a function on some incoming arc of a transition, it would be necessary to invert the function to find tokens to assign to the variable such that the arc expression evaluates to some token marking the input place connected to the transition through the arc.

2.14 Timed Token

The waiting of tokens in places is achieved by attaching a natural number to each token marking a place. Such a number expresses, in some time unit, the (discrete) amount of time the token must wait before it can be removed from the place. These natural numbers are meant to be decremented as time passes. The following definition formalizes tokens with natural numbers attached.

Definition 14. Given a token taxonomy $\boldsymbol{X}$, the family $\{TK_y\}_{y \in Y}$ of *timed tokens* (*T-tokens*) is defined as follows:

$$\forall\, y \in Y : TK_y = \{\ k{:}\theta \mid k \in K_y \wedge \theta \in \mathbf{N}\ \}.$$

The set of all T-tokens is thus

$$TK = \bigcup_{y \in Y} TK_y\ .$$

If $k{:}\theta \in TK$, θ is called *waiting time* of $k{:}\theta$.
If $k{:}\theta \in TK$ and $y \in Y$, we say that $k{:}\theta$ *has type* y iff

$$k{:}\theta \in TK_y.$$

T-tokens are organized as a family indexed by types, in complete analogy to tokens. In fact, the types of a T-token coincide with those of the constituent token.

2.15 Marking

While in an SCP-net a marking associates a multiset of tokens to each place, in an ESCP-net a marking associates a multiset of T-tokens to each place (all having the type labeling the place, of course), as expected.

Definition 15. Given an ESCP-net ***ESCPN***, a *marking* for ***ESCPN*** is a function

$$\mu \in [\ P.p \rightarrow \mathcal{M}_{\omega}(TK_{\psi(p)})\ ].$$

M is the set of all markings.

The marking of the ESCP-net in Figure 1 is indicated by the tokens inside places (we have in fact omitted waiting times, which are all null, to fit strings in circles; for the same reason, we have used abbreviations for base token symbols). The time unit for the ESCP-net is assumed to be 1 minute.

[7] Even if there were only a finite number of tokens of type **R**, as in practical implementations, in general they would still be too many to manage (e.g. as many as floating point numbers in some 32-bit format).

2.16 Enabling and Firing

Transition firings in ESCP-nets are analogous to those in SCP-nets. The difference is that errors and waiting times must be taken into account. Concerning errors, a transition is defined as being not enabled with a binding in case the guard, a surrounding arc expression, or a stochastic time of an output place evaluates to **E**. For what concerns waiting times, only T-tokens with null waiting times can be removed from a place. When a T-token must be added to a place, its (initial) waiting time is randomly generated from the t.p.f. which the stochastic time labeling the place evaluates to. The following definition formalizes these concepts.

Definition 16. Given an ESCP-net ***ESCPN***, let:
(1) $\mu \in M$;
(2) $t \in T$;
(3) β be a binding for $Var(t)$.
We say that t is *enabled* with β in μ iff:
(1) $\forall\, a \in In(t) \cup Out(t) : \beta(\varepsilon(a)) \neq \mathbf{E}$;
(2) $\forall\, a \in Out(t) : \beta(\tau(\chi_P(a))) \neq \mathbf{E}$;
(3) $\beta(\gamma(t)) = \mathsf{T}$;
(4) $\forall\, p \in P : \bigcup_{a \in In(t) \cap Out(p)} \{\, \beta(\varepsilon(a)):0 \,\}_{\mathsf{m}} \subseteq \mu(p)$.

If t is enabled with β in μ, and if

$$\zeta \in [\, Out(t) \rightarrow \{\, z \in \mathbf{R} \mid 0 \leq z < 1 \,\} \,],$$

the marking μ' produced by *t firing* with β and ζ in μ is defined as follows:

$$\forall\, p \in P : \mu'(p) = \left(\mu(p) - \bigcup_{a \in In(t) \cap Out(p)} \{\, \beta(\varepsilon(a)):0 \,\}_{\mathsf{m}} \right) \cup \bigcup_{a \in Out(t) \cap In(p)} \{\, \beta(\varepsilon(a)):\Theta(a) \,\}_{\mathsf{m}},$$

where

$$\forall\, a \in Out(t),\ \beta(\tau(\chi_P(a))) = \{\pi_i\}_{i \in \mathbf{N}} : \Theta(a) = \min \{\, i \in \mathbf{N} \mid \sum_{h \in \{0,\ldots,i\}} \pi_h > \zeta(a) \,\}.$$

In order to formalize the random generation of waiting times for tokens added to the output places of t, we have defined the firing with respect to a non-negative real number less than 1, $\zeta(a)$, associated to each outgoing arc a of t, which represents a uniformly random value. This real number is "mapped" to a natural number $\Theta(a)$ according to the evaluated t.p.f. $\{\pi_i\}_{i \in \mathbf{N}}$ of the corresponding output place: $\Theta(a)$ is the smallest natural number such that $\pi_0 + \ldots + \pi_{\Theta(a)}$ is greater than $\zeta(a)$. In this way, if $\zeta(a)$ is uniformly random, $\Theta(a)$ is random according to the t.p.f. $\{\pi_i\}_{i \in \mathbf{N}}$.

For instance, in the ESCP-net in Figure 1, consider the transition START and the binding β for Var(START) such that

$$\beta(pc1) = \mathsf{magenta},$$
$$\beta(pc2) = \mathsf{yellow},$$
$$\beta(shp) = \mathsf{cube},$$
$$\beta(dim) = 2,$$
$$\beta(q1) = 80,$$
$$\beta(q2) = 103.$$

We have that START is enabled with β, and the firing has the effect of removing cube;2;none from UNPAINTED, adding cube;2;red to PAINTING, changing magenta;80 in TANKS1 to magenta;72.8, and changing yellow;103 in TANKS2 to yellow;95.8. The waiting times of the tokens of the new marking are all 0, except the

token in PAINTING, whose waiting time is between 36 and 48 minutes.

In general, the firing of START represents the start of a painting processes of a cube or sphere, which consumes quantities of paint proportional (according to a factor 0.3 liters per square meter) to the surface area of the cube or sphere, and which takes a uniformly random time between two values also proportional (according to factors 1.5 and 2 minutes per square meter) to the surface area of the cube or sphere. END can fire only after the waiting time of the token in PAINTING has become null. Thus, stochastic times allow different tokens to wait different amounts of time in a same place.

3. Future Work

The main direction of future work consists in the study of formal analysis (and synthesis as well) methods for ESCP-nets. This work is closely related to that for SCP-nets. We have in fact already discovered some formal results which relate some properties of ESCP-nets to analogous properties of SCP-nets. For instance, for each ESCP-net there exist an SCP-net and a mapping from the markings of the ESCP-net to those of the SCP-net, such that for each firing in the ESCP-net there is a corresponding firing in the SCP-net which preserves the correspondence (according to the mapping) between ESCP-net markings and SCP-net markings. This implies that boundedness of the SCP-net is sufficient for boundedness of the ESCP-net, and reversibility of the SCP-net is necessary for reversibility of the ESCP-net. So, formal analysis carried upon the SCP-net can give useful information about properties of the ESCP-net. We have not presented these formal results here because they are still under study and we prefer to give a more systematic account in the future.

Although ESCP-nets have arisen in the field of plant simulation, and presently we have employed them in this field only, we think that their characteristics can be well-suited to other fields as well. So, we have planned to investigate other fields in which ESCP-nets might be successfully employed.

Appendix

In this appendix we explain the mathematical concepts and notations used in Section 2.

$\mathbf{B} = \{\mathsf{T}, \mathsf{F}\}$ is the set of booleans.

$\mathbf{N} = \{0, 1, 2, \ldots\}$ is the set of natural numbers.

$\mathbf{R}$ is the set of real numbers. If $x \in \mathbf{R}$, Round(x) is the integer number obtained by rounding x (down if the fraction part is less than 0.5, up otherwise).

If $A_1, \ldots, A_n$ ($n \geq 2$) are sets, $A_1 \times \ldots \times A_n$ is the cartesian product of $A_1, \ldots, A_n$, i.e. the set of all n-tuples $\langle a_1, \ldots, a_n \rangle$ where $a_1 \in A_1, \ldots, a_n \in A_n$.

If A and B are sets, $[A \to B]$ is the set of all (total) functions with domain A and codomain B.

If A and B are sets, and $B(a) \subseteq B$ is a subset of B for each $a \in A$, $[A.a \to B(a)]$ is the set of all functions $f \in [A \to B]$ such that $f(a) \in B(a)$ for each $a \in A$.

If A and B are sets, and $\delta_a \in B$ is an element of B for each $a \in A$, $\{\delta_a\}_{a \in A}$ is the family of elements of B indexed by the elements of A, i.e. the function $f \in [A \to B]$ such that $f(a) = \delta_a$ for each $a \in A$. If $A = A' \times A'$ for some set A', we just write $\{\delta_{a_1,a_2}\}_{a_1,a_2 \in A'}$ instead of $\{\delta_{\langle a_1,a_2 \rangle}\}_{\langle a_1,a_2 \rangle \in A}$.

If $B = \{B_a\}_{a \in A}$ and $C = \{C_a\}_{a \in A}$ are families of pairwise disjoint sets, $[B_a \to C_a]_{a \in A}$ is the set of all functions $f \in \left[\bigcup_{a \in A} B_a \to \bigcup_{a \in A} C_a\right]$ such that $f(b) \in C_a$ for each $a \in A$ and $b \in B_a$.

If A is a set, $\mathcal{P}_\omega(A)$ is the set of all finite subsets of A.

If A is a set, $\mathcal{M}_\omega(A)$ is the set of all finite multisets over A, where a finite multiset over A is a function $ms \in [A \to \mathbf{N}]$ such that $\sum_{a \in A} ms(a) \in \mathbf{N}$. If $a \in A$, $ms(a)$ is the multiplicity of a in ms. If $a \in A$, $\{a\}_\mathsf{m}$ is the multiset $ms \in \mathcal{M}_\omega(A)$ defined by $ms(a) = 1$, and $ms(a') = 0$ for each $a' \in A - \{a\}$. If $ms_1, ms_2 \in \mathcal{M}_\omega(A)$, $ms_1 \cup ms_2$ is the multiset $ms \in \mathcal{M}_\omega(A)$ defined by $ms(a) = ms_1(a) + ms_2(a)$ for each $a \in A$. If $ms_1, ms_2 \in \mathcal{M}_\omega(A)$, we write $ms_1 \subseteq ms_2$ to express that $ms_1(a) \leq ms_2(a)$ for each $a \in A$. If $ms_1, ms_2 \in \mathcal{M}_\omega(A)$, and $ms_1 \subseteq ms_2$, $ms_2 - ms_1$ is the multiset $ms \in \mathcal{M}_\omega(A)$ defined by $ms(a) = ms_2(a) - ms_1(a)$ for each $a \in A$.

If A is a set, a directed graph of elements of A is a pair $\Gamma = \langle N, E \rangle$ where $N \subseteq A$ is the set of nodes, and $E \subseteq N \times N$ is the set of edges. Γ is empty iff $N = \varnothing$. Γ is finite iff N is finite. If $a, a' \in N$, we write $a \to_\Gamma a'$ to express that there are $a_1, \ldots, a_n \in N$ ($n \geq 0$) such that $\langle a, a_1 \rangle, \langle a_1, a_2 \rangle, \ldots, \langle a_n, a' \rangle \in E$ (i.e. there is a path from a to a' in Γ). Γ is acyclic iff no $a \in N$ is such that $a \to_\Gamma a$; we write "DAG" as a shortcut for "directed acyclic graph". A node $a \in N$ is terminal iff no $a' \in N$ is such that $a' \to_\Gamma a$ (note that if Γ is acyclic, there is at least one terminal node). Term(Γ) is the set of all terminal nodes of Γ, and NTerm(Γ) is the set of all non-terminal nodes of Γ (i.e. NTerm(Γ) = N − Term(Γ)). A node $a \in N$ is isolated iff no $a' \in N$ is such that $a' \to_\Gamma a$ or $a \to_\Gamma a'$ (note that an isolated node is also terminal).

References

[1] A. Camurri, A. Coglio, "A Petri Net-based Architecture for Plant Simulation", in *Proceedings of the 6th IEEE Conference on Emerging Technologies and Factory Automation*, UCLA, Los Angeles, California (USA), September 1997.

[2] A. Camurri, A. Coglio, "Extended Simple Colored Petri Nets: A Tool for Plant Simulation", in *Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics*, Hyatt Orlando, Orlando, Florida (USA), October 1997.

[3] A. Camurri, A. Coglio, *Simple Colored Petri Nets*, Technical Report, DIST, University of Genoa, Italy, November 1997, available at `ftp://ftp.dist.unige.it/pub/infomus/Publications/scpnets.ps.zip`.

[4] A. Camurri, A. Coglio, *Specification of an Executor of Extended Simple Colored Petri Nets*, Technical Report, DIST, University of Genoa, Italy, December 1997, available at `ftp://ftp.dist.unige.it/pub/infomus/Publications/specexec.ps.zip`.

[5] K. Jensen, "Coloured Petri Nets: A High Level Language for System Design and Analysis", in G. Rozenberg (ed.), *Advances in Petri nets 1990*, Lecture Notes in Computer Science, vol. 483, Springer-Verlag, pp. 342–416, 1990.

[6] T. Murata, "Petri Nets: Properties, Analysis and Applications", in *Proceedings of the IEEE*, 77(4), pp. 541-580, April 1989.

[7] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice Hall, Englewood Cliffs, NJ, 1981.